IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: M. GIRKAR *et al.* Attorney Docket No.: 19111.0228

Application No.: 09/852,008 Group Art Unit: 2177

Filed: May 10, 2001 Examiner: D. Le

For: DISASTER RECOVERY WITH BOUNDED DATA LOSS

SUBSTITUTE APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

Sir:

This Appeal Brief is submitted in support of the Notice of Appeal dated September 10, 2004 and the Notification of Non-Compliant Appeal Brief Dated January 9, 2007.

I. REAL PARTY IN INTEREST

Oracle International Corp. is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals and interferences.

III. STATUS OF THE CLAIMS

Claims 1-16 are pending in this appeal, in which no claim has earlier been cancelled. No claim is allowed. This appeal is therefore taken from the final rejection of claims 1-16 on March 16, 2004.

IV. STATUS OF AMENDMENTS

No amendment to claims has been filed after final rejection.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The present invention addresses difficulties in database systems. For example, the Background section of the present application explains that "it is difficult to characterize the amount of data lost in terms that database owners can best understand. The maximum exposure for loss of data in this approach is usually described in terms of the size of the redo logs, but this information is not helpful for database owners, who would rather want to know how many orders were lost" (Background, ¶ 7). In recognition of this problem, the method recited in the claims sets for the use of "a predetermined number of transactions" in synchronizing a transaction (claim 1; see FIG. 2, items 203, 211 FIG. 3, items 303, 311 showing the use of a transaction counter). "Because the predetermined bound is specified in terms of the number of transactions, the database operator can set a meaningful tradeoff between performance and data availability that is appropriate for the particular needs of the database operator's installation" (Summary, ¶ 11).

Methods and systems consistent with the present invention may allow a database operator to set a bound that limits the number of transactions that can be lost. To improve performance over the synchronous approach, transactions are placed in a buffer (*see, e.g.*, FIG. 1, item 121, ¶ 29) to be sent to the standby system, but, to limit the data loss to a predetermined number of transactions, the transactions are synchronized based on the predetermined bound and on the number of transactions currently in the buffer. (*See, e.g.*, specification, ¶ 11, claim 1, claim 16).

According to one aspect, the present invention further comprises the step of executing a net server process to transmit the transaction over a network connection to the standby database system. Then, an acknowledgement is receives indicating that a redo record for the transaction has been written to a standby log at the standby database system. Finally, the transaction is removed from the buffer in response to the acknowledgment. (*See, e.g.*, specification, ¶ 15, claim 7).

The step of synchronizing may include the steps of storing a counter indicating a number of transactions in the buffer. When adding the transaction to the buffer, the counter is incremented. When removing the transaction from the buffer, the counter is decremented. A commit of the transaction is blocked when the counter is not less than the predetermined number of transactions. Finally, the commit of the transaction is acknowledged when the counter is less than the predetermined number of transactions. (*See, e.g.*, specification, ¶ 12, claim 9).

One aspect of methods and systems consistent with the present invention relates to a method and software for replicating data of a primary database system. Accordingly, a buffer of transactions to be sent to a standby database system is maintained, and a transaction performed on the primary database system is synchronized based on a number of transactions in the buffer and a predetermined number of transactions. For example, a commit of the transaction is blocked until the number of transactions in the buffer is less than the predetermined number of transactions. (*See, e.g.,* specification, ¶ 12, claim 11, claim 12, ¶ 31, FIG. 2, item 203).

In one embodiment, the synchronization is enforced by the log writer process (*see, e.g.,* ¶ 29, ¶ 31, FIG. 1, item 111), but, in another embodiment, the synchronization is handled by user process before submitting the transaction to the log writer process. Determining the number of transactions in the buffer can be done by inspecting (or "sniffing") the buffer or by maintaining a counter that is incremented when the log writer process submits a transaction in the buffer and decremented when a net server process receives an acknowledgement that the redo for the transaction has been written to the standby logs at the standby database system. (*See, e.g.,* specification, ¶ 13, ¶31-32, ¶ 36, FIGS. 2, 3, claim 11, claim 12, claim 14).

Another aspect of methods and systems consistent with the present invention pertains to a method and software for operating a log writer process to replicate data of a primary database system. In particular, the log writer process is programmed to record a transaction in a redo log and compare a counter indicating a number of the transactions in a queue of transactions to be sent to a standby database system and a predetermined bound of transactions. If the counter is greater than the predetermined bound, then the log writer process blocks a commit of the transaction until the counter is less than the predetermined bound. On the other hand , if the counter is less than the predetermined bound, then the log writer process increments the counter and acknowledges the commit of the transaction. (*See, e.g.,* specification, ¶ 14, ¶¶ 29-31, ¶ 36, FIGS. 2, 3, claim 11, claim 12).

Still another example involves a method and software for operating a net server process to replicate data of a primary database system. Specifically, the net server process is configured to access a transaction maintained in a buffer of transactions to be sent to a standby database system, transmit the transaction over a network connection to the standby database system, receive an acknowledgement that the transaction has been committed at the standby database

system; and, in response to the acknowledgment, remove the transaction from the queue and decrement the counter. (*See, e.g.,* specification ¶ 15, FIGS. 2, 3, ¶ 29, ¶¶ 32-33, claim 14).

Methods and systems consistent with the present invention can be readily applied to a parallel database server environment in which the parallel database servers have shared disk access to the primary database. In such an environment, each database server can be given its own predetermined bound independent of the other database servers, (for example, by dividing the database operator's bound by the number of database servers), or the parallel database server can use a shared disk counter of transactions. (*See, e.g.,* specification, ¶ 16, ¶¶ 38-39, claim 16).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether claims 1-6, 8, and 10 are obvious under 35 U.S.C. § 103(a) based on *Rastogi et al.* (U.S. 6,205,449) in view of *Cooper et al.* (U.S. 6,079,000).

Whether claims 7 and 9-15 are obvious under 35 U.S.C. § 103(a) based on *Rastogi et al.* in view of *Cooper et al.* and further in view of *Hapner et al.* (U.S. 5,940,827).

Whether claim 14 is obvious under 35 U.S.C. § 103(a) based on *Rastogi et al.* in view of *Hapner et al.*

Whether claim 16 is obvious under 35 U.S.C. § 103(a) based on *Rastogi et al.* in view of *Cooper et al.* and further in view of *Nilsen et al.* (U.S. 5,668,986).

VII. ARGUMENT

A. ALL PENDING CLAIMS ARE NON-OBVIOUS OVER *RASTOGI ET AL.* BECAUSE NONE OF THE REFERENCES TEACH OR SUGGEST SYNCHRONIZING TRNASACTIONS BASED ON A "PREDETERMINED NUMBER OF TRANSACTIONS."

The initial burden of establishing a *prima facie* basis to deny patentability to a claimed invention under any statutory provision always rests upon the Examiner. *In re Mayne,* 104 F.3d 1339, 41 USPQ2d 1451 (Fed. Cir. 1997); *In re Deuel,* 51 F.3d 1552, 34 USPQ2d 1210 (Fed. Cir. 1995); *In re Bell,* 991 F.2d 781, 26 USPQ2d 1529 (Fed. Cir. 1993); *In re Oetiker,* 977 F.2d 1443, 24 USPQ2d 1443 (Fed. Cir. 1992). In rejecting a claim under 35 U.S.C. § 103, the Examiner is required to provide a factual basis to support the obviousness conclusion. *In re Warner,* 379

F.2d 1011, 154 USPQ 173 (CCPA 1967); *In re Lunsford,* 357 F.2d 385, 148 USPQ 721 (CCPA 1966); *In re Freed,* 425 F.2d 785, 165 USPQ 570 (CCPA 1970).

1. CLAIMS 1-6, 8, AND 10 ARE NON-OBVIOUS OVER *RASTOGI ET AL.* IN VIEW OF *COOPER ET AL.*

Reversal of the rejection of claim 1-6, 8, and 10 over *Rastogi et al.* and *Cooper at al.* is respectfully requested because the references do not teach or otherwise suggest the limitations of the claims. For example, independent claim 1 recites: "synchronizing a transaction performed on the primary database system based on a number of transactions in the buffer and a **predetermined number of transactions**."

The Examiner correctly acknowledges that "Rastogi does not explicitly teach a predetermined number of transactions" (p. 3). Indeed, the portion of Rastogi *et al.* cited for transactions, col. 8:3-, only discusses "transactions" in the model described in the reference. There is no mention or suggestion of "synchronizing a transaction," much less "synchronizing a transaction" based on "a predetermined number of transactions."

Cooper et al. too fails to show this feature. In fact, *Cooper et al.* exhibits many of the same difficulties described in the background and addressed by the invention recited in claim 1. For example, *Cooper et al.* recommends managing the "optimal transfer efficiency" between the audit host memory **342** and the XPC cache area **350** based on a "predetermined size of audit host memory **342**," which is given in terms of a "predetermined number of address locations within audit host memory **342**" (col. 12:39-40). Referring now to FIG. 9 of *Cooper et al.,* transactions **344, 354, 362,** and **370** clearly have different sizes in terms of the number of audit host memory **342** locations. For example, transactions **354** is about half the size of transactions **344** and about a third of the size of transaction **362**. When transaction sizes are so variable as in *Cooper et al.,* pre-specifying buffers in terms of number of memory locations or bytes does not meaningfully specify a predetermined number of transactions" as set forth in independent claim 1. In fact, this variability in transaction size is what dooms *Cooper et al.'s* approach to exhibit the problems addressed by the invention of independent claim 1.

The portion of *Cooper et al.* cited in the Office Action, col. 12:30-43, does not support the rejection. Although *Cooper et al.* speculates that "the optimal transfer characteristics of the physical audit trail **378** may determine when a sufficient number of transactions have been

accumulated in audit host memory **342**" (col. 12:33-36), *Cooper et al.* then defines what it means a "sufficient number of transactions"- not in terms of a "predetermined number of transactions" as recited in claim 1 - but explicitly in terms of the size of audit host memory **342**: "Thus, the optimal transfer efficiency may correspond to the a predetermined size of audit host memory **342**" (col. 12:36-38). Due to the variability in transaction sizes evident in FIG. 9, *Cooper et al.'s* criterion based on a predetermined memory size can only crudely and inaccurately correspond to the actual number of transactions in the audit host memory **342**. However, *Cooper et al.* is not concerned with providing a meaningful bound to database administrators, but focused on transferring data between memories as efficiently as possible. Thus, *Cooper et al.* actually **teaches against** using a "predetermined number of transactions," since transferring one predetermined number of tiny transactions is less efficient than transferring the same predetermined number of large transactions. If a proposed modification would render the reference being modified unsatisfactory for its intended purpose, then there is no suggestion or motivation to make the proposed modification. *In re Gordon,* 733 F.2d 900, 221 USPQ 1125 (Fed. Cir. 1984).

As for col. 14:41-43, also cited in the Office Action, *Cooper et al.* makes a similar recommendation for the size of the XPC cache area **350** in terms of "a predetermined number of address locations within XPC cache area 350" (col. 14:39-41) and is therefore similarly deficient for the same reasons as with the size of audit host memory **342**. Furthermore, *Cooper et al.* states that the "synchronous audit data request results in a portion of the audit host memory **342** being written to a corresponding portion of non-volatile memory storage such as XPC cache **350**" (col. 11:50-53), thus the relevance of the size of the XPC cache area **350** to "synchronizing a transaction" as recited in independent claim **1** is not immediately apparent.

2. CLAIMS 7, 9, AND 11-16 ARE NON-OBVIOUS OVER *RASTOGI ET AL.* AND *COOPER ET AL.* IN VIEW OF *HAPNER ET AL.*

Concerning dependent claims 7 and 9 as well as claims 11-16, the use of the non-analogous *Hapner et al.* does not support the rejection by curing the deficiencies of *Rastogi et al.* and *Cooper et al.* or by disclosing the additional features recited in claims 7, 9, and 11-16. *Hapner et al.* relates to maintaining a database cache in conjunction with a persistent database portion and uses a "transaction counter" (col. 3:44) for keeping track of how many open

transactions there are in the database cache **140.** *Hapner et al.* lacks any disclosure of using such a "transaction counter" for any set of "transactions to be sent to a standby database system." Since neither *Rastogi et al.* nor *Cooper et al.* operate with a predetermined number of transactions to be sent to a standby database system, there is no motivation to modify *Rastogi et al.* and/or *Cooper et al.* to count something none of the references seem to care about.

Furthermore, the only comparison of *Hapner et al.'s* transaction counter is with zero (0) in FIG. 10, step 469, and FIG. 11, steps 512 and 536. However, claim 11 explicitly recites "**compare** the counter and **the predetermined bound**" and claim 12 recites "**comparing** a counter indicating a number of the transactions in a queue of transactions to be sent to a standby database system and **a predetermined bound of transactions**." Whatever *Cooper et al.* may be thought to disclose about the optimal transfer size of audit host memory **342**, the optimal transfer size certainly cannot be zero! Thus, even if the Examiner might be correct in responding that "Hapner does apply a transaction counter in a replicating data" (p. 11), the claims are more specific than that and the use of a counter in the specific context of claims 7, 9, and 11-16 is not disclosed in *Hapner et al.*

3. CLAIM 14 IS NON-OBVIOUS OVER *RASTOGI ET AL.* IN VIEW OF *HAPNER ET AL.*

Claim 14 was further rejected over *Rastogi et al.* in view of *Hapner et al.* However, as argued above in section VII. A. 2, claim 14 is not obvious over *Rastogi et al.* and *Cooper et al.* in view of *Hapner et al.* because all three references lack a teaching or suggestion of using a "transaction counter" for any set of "transactions to be sent to a standby database system." Similarly, they also lack a teaching or suggestion of "transmitting the transaction over a network connection" to a standby database system, and "removing the transaction" and "decrementing a counter" as recited by claim 14. Accordingly, claim 14 is also not obvious over the smaller set of these references, *i.e., Rastogi et al.* in view of *Hapner et al.*

4. CLAIM 16 IS NON-OBVIOUS OVER *RASTOGI ET AL.* FURTHER IN VIEW OF *NILSEN ET AL.*

Nilsen et al., applied only against claim 16, does not furnish a disclosure of "synchronizing a transaction performed on the primary database system based on a number of

transactions in the buffer and the corresponding bound" which is missing in *Rastogi* and *Cooper et al.* as explained in section VII. A. 2. The Examiner, properly, did not rely on *Nilsen et al.* for this factual inadequacy of *Rastogi et al.* and *Cooper et al.* Thus, claim 16 too is patentable over *Rastogi et al., Cooper et al.,* and *Nilsen et al.,* and its allowance is respectfully requested.

VIII. CONCLUSION AND PRAYER FOR RELIEF

For the foregoing Reasons, Appellants request the Honorable Board to reverse each of the Examiner's rejections.

Respectfully submitted,
BINGHAM MCCUTCHEN LLP

Dated: July 9, 2007

By: 

Siddhesh V. Pandit, Registration No. 58,572
BINGHAM MCCUTCHEN LLP
2020 K Street, NW
Washington, D.C. 20006
(202) 373-6513 Telephone
(202) 373-6001 Facsimile

IX. CLAIMS APPENDIX

1. (Original) A method for replicating data of a primary database system, comprising the steps of:

maintaining a buffer of transactions to be sent to a standby database system; and

synchronizing a transaction performed on the primary database system based on a number of transactions in the buffer and a predetermined number of transactions.

2. (Original) A method according to claim 1, wherein the step of synchronizing includes the step of:

blocking a commit of the transaction until the number of transactions in the buffers is in a predetermined numerical relationship with the predetermined number of transactions.

3. (Previously Presented) A method according to claim 2, wherein:

said blocking the commit of the transaction until the number of transactions in the buffers is in the predetermined numerical relationship with the predetermined number of transactions including blocking the commit of the transaction until the number of transactions in the buffers is less than the predetermined number of transactions.

4. (Original) A method according to claim 1, further comprising the step of:

executing a log writer process to record the transaction in a redo log.

5. (Original) A method according to claim 4, wherein:

the log writer process performs the step of synchronizing.

6. (Original) A method according to claim 4, wherein:

a database application process performs the step of synchronizing before submitting the transaction to the log writer process.

7. (Original) A method according to claim 1, further comprising the step of:

executing a net server process to:

transmit the transaction over a network connection to the standby database system,

receive an acknowledgement that a redo record for the transaction has been written to a standby log at the standby database system, and

remove the transaction from the buffer in response to the acknowledgement.

8. (Original) A method according to claim 1, further comprising the steps of:

receiving input from an operator indicating a transaction loss bound; and

setting the predetermined number of transactions based on the transaction loss bound.

9. (Original) A method according to claim 1, wherein the step of synchronizing includes the steps:

storing a counter indicating a number of the transactions in the buffer;

when adding the transaction to the buffer, incrementing the counter;

when removing the transaction from the buffer, decrementing the counter;

blocking a commit of the transaction when the counter is not less than the predetermined number of transactions; and

acknowledging the commit of the transaction when the counter is less than the predetermined number of transactions.

10. (Original) A computer-readable medium bearing instructions for causing one or more processors to performs the steps of the method according to claim 1.

11. (Original) A method for replicating data of a primary database system, comprising the steps of:

maintaining a queue of transactions to be sent to a standby database system;

storing a counter indicating a number of the transactions in the queue;

storing a predetermined bound of transactions;

executing a log writer process to:

record the transaction in a redo log,

compare the counter and the predetermined bound,

if the counter is not less than the predetermined bound, then block a commit of the transactions until the counter is less than the predetermined bound, and

if the counter is less than the predetermined bound, then increment the counter and acknowledge the commit of the transaction; and

executing a net server process to:

transmit the transaction over a network connection to the standby database system,

receive an acknowledgement that a redo record for the transaction has been written to a standby log at the standby database system, and

in response to the acknowledgement, remove the transaction from the queue and decrement the counter.

12. (Original) A method for operating a log writer process to replicate data of a primary database system, comprising the steps of:

recording a transaction in a redo log;

comparing a counter indicating a number of the transactions in a queue of transactions to be sent to a standby database system and a predetermined bound of transactions;

if the counter is not less than the predetermined bound, then blocking a commit of the transaction until the counter is less than the predetermined bound, and

if the counter is less than the predetermined bound, then incrementing the counter and acknowledging the commit of the transaction.

13. (Original) A computer-readable medium bearing instructions for causing one or more processors to perform the steps of the method according to claim 12.

14. (Previously Presented) A method for operating a net server process to replicate data of a primary database system, comprising the steps of:

accessing a transaction maintained in a buffer of transactions to be sent to a standby database system;

transmitting the transaction over a network connection to the standby database system;

receiving an acknowledgment that a redo record for the transaction has been written to a standby log at the standby database system, and

in response to the acknowledgement, removing the transaction from the queue and decrementing a counter.

15. (Original) A computer-readable medium bearing instructions for causing one or more processors to perform the steps of the method according to claim 14.

16. (Original) A method for replicating data in a primary database system having multiple database servers operating in parallel and accessing a common database on a shared disk, said method comprising the steps of:

setting a bound for each of the multiple database servers;

for each of the multiple database servers, performing the steps of:

maintaining a buffer of transactions to be sent to a standby database system; and

synchronizing a transaction performed on the primary database system based on a number of transactions in the buffer and the corresponding bound.

X. EVIDENCE APPENDIX

None.

XI. RELATED PROCEEDINGS APPENDIX

None.